readme

文档

项目详细文档见 doc

运行

将整个项目作为工程文件打开，使用 Clion 编译，编译后将 cmake-build-debug/ 目录下的 `jpp.exe` 文件放入当前目录下。

或直接运行当前目录下的 jpp.exe 文件。运行产生文件见 项目结构

项目结构

- data
  - action_table.csv 动作表
  - goto_table goto.csv 表
  - input_code.csv 测试源码
  - inst.asm 输出的汇编指令
  - kw.txt 系统内置关键字及编码
  - quad.txt 四元组序列
  - reduce.txt 输出的归约式序列
  - sign.txt 生成的符号表，调试使用
  - token.txt 生成的 token 表，调试使用
  - CG_data 汇编代码生成器测试数据
  - LA_data 词法分析器测试数据
  - LR_data 语法分析器与中间代码生成器测试数据
- LA 词法分析器代码
- LR 语法分析器代码
- CG 汇编代码生成器代码
- doc.md 文档
- main.cpp 主函数

一些说明

1. 为了更好的实现词法分析器的功能，系统内置字符串（关键字、运算符等）也会被放入符号表
2. LA/LA_test, LR/LR_test 为测试文件，在其中编写了测试函数

文法

```
[非终结符]
SENTENCE STATE TYPE IDS ASSIGN EXPR TT FF SENTENCES

[终结符]
semic int real id + * ( ) = , dec hex oct bin realc
```

```
[文法起始符]
SS

[生成式]
SS -> SENTENCES;
SENTENCES -> SENTENCES SENTENCE;
SENTENCES -> SENTENCE;
SENTENCE -> STATE semic;
SENTENCE -> ASSIGN semic;

STATE -> TYPE IDS;
TYPE -> int;
TYPE -> real;
IDS -> IDS , id;
IDS -> id;

ASSIGN -> id = EXPR;
EXPR -> EXPR + TT;
EXPR -> TT;
TT -> TT * FF;
TT -> FF;
FF -> ( EXPR );
FF -> id;
FF -> dec;
FF -> hex;
FF -> oct;
FF -> bin;
FF -> realc;
```

实现了基本的声明语句与赋值语句，但**不支持声明的时候赋值，所有语句末尾需要填写 `semic` 作为结束符**

实例

```
int a, b semic
real c, d semic
a = 1 semic
b = 2 semic
c = a + b semic
d = a * b semic
```

添加了语义动作之后的文法

```
[非终结符]
SENTENCE STATE TYPE IDS ASSIGN EXPR TT FF SENTENCES

[终结符]
```

```
semic int real id + * ( ) = , dec hex oct bin realc

[文法起始符]
SS

[生成式]
SS -> SENTENCES;
SENTENCES -> SENTENCES SENTENCE;
SENTENCES -> SENTENCE;
SENTENCE -> STATE semic;
SENTENCE -> ASSIGN semic;

STATE -> TYPE M IDS;
M -> ; {M.value_type = TYPE.value_type; M.width = TYPE.width}
TYPE -> int;  {TYPE.value_type = INT; TYPE.width = 4;}
TYPE -> real; {TYPE.value_type = REAL; TYPE.width = 4;}
IDS -> IDS, N id; {enter(id.name, id.value_type, id.width, offset); offset
+= id.width; }
N -> ; {N.value_type = IDS.value_type; N.width = IDS.width;}
IDS -> P id;  {enter(id.name, id.value_type, id.width, offset); offset +=
id.width; }
P -> ; {P.value_type = IDS.value_type; P.width = IDS.width;}

ASSIGN -> id = EXPR;  {addr = id.name; if p != NULL then gencode("=",
addr, EXPR.addr, -1); else error;}
EXPR -> EXPR + TT;   {EXPR.addr = newtemp(); gencode("+", EXPR.addr,
EXPR1.addr, TT.addr);}
EXPR -> TT;           {EXPR.addr = TT.addr;}
TT -> TT * FF;       {TT.addr = newtemp(); gencode("*", TT.addr, TT1.addr,
FF.addr);}
TT -> FF;             {TT.addr = FF.addr;}
FF -> ( EXPR );       {FF.addr = EXPR.addr;}
FF -> id;             {FF.addr = id.name;}
FF -> dec;            {FF = newtemp(); gencode("=", FF.addr, dec);}
FF -> hex;            {FF = newtemp(); gencode("=", FF.addr, hex);}
FF -> oct;            {FF = newtemp(); gencode("=", FF.addr, hex);}
FF -> bin;            {FF = newtemp(); gencode("=", FF.addr, bin);}
```

由于生成式为读取动作表式产生，并且放入哈希表，无法确认位置，因此决定在每个非终结元素后边加上动作，
当归约出该符号时，根据调用的归约式进行判断，进而选择使用哪个函数来执行动作。